In [ ]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```python
df = pd.read_csv('data/NCI60_X.csv',index_col=0)
df.head(10)
```

Out[ ]:

	1	2	3	4	5	6	7	8	9	10	...	6821
V1	0.300000	1.180000	0.550000	1.140000	-0.265000	-7.000000e-02	0.350000	-0.315000	-0.450000	-0.654980	...	-0.990020
V2	0.679961	1.289961	0.169961	0.379961	0.464961	5.799610e-01	0.699961	0.724961	-0.040039	-0.285019	...	-0.270058
V3	0.940000	-0.040000	-0.170000	-0.040000	-0.605000	0.000000e+00	0.090000	0.645000	0.430000	0.475019	...	0.319981
V4	0.280000	-0.310000	0.680000	-0.810000	0.625000	-1.387779e-17	0.170000	0.245000	0.020000	0.095019	...	-1.240020
V5	0.485000	-0.465000	0.395000	0.905000	0.200000	-5.000000e-03	0.085000	0.110000	0.235000	1.490019	...	0.554980
V6	0.310000	-0.030000	-0.100000	-0.460000	-0.205000	-5.400000e-01	-0.640000	-0.585000	-0.770000	-0.244980	...	-0.590020
V7	-0.830000	0.000000	0.130000	-1.630000	0.075000	-3.600000e-01	0.100000	0.155000	-0.290000	-0.084981	...	0.189980
V8	-0.190000	-0.870000	-0.450000	0.080000	0.005000	3.500000e-01	-0.040000	-0.265000	-0.310000	-0.244980	...	-0.210019
V9	0.460000	0.000000	1.150000	-1.400000	-0.005000	-7.000000e-01	-0.920000	-0.515000	-0.280000	-0.114980	...	0.089980
V10	0.760000	1.490000	0.280000	0.100000	-0.525000	3.600000e-01	0.600000	0.175000	0.580000	1.145019	...	0.299980

10 rows × 6830 columns

In [ ]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 64 entries, V1 to V64
Columns: 6830 entries, 1 to 6830
dtypes: float64(6830)
memory usage: 3.3+ MB
```

In [ ]:
```python
x_raw = df.copy()
```

In [ ]:
```python
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
```

Standard scaling all columns

In [ ]:
```python
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x_raw)
```

creating 4 kmeans clusters

In [ ]:
```python
kmc = KMeans(n_clusters=4)
kmc.fit(x_scaled)
```

Out[ ]:
```
▾ KMeans
KMeans(n_clusters=4)
```

creating 4 agglomerative clusters

In [ ]:
```python
agg = AgglomerativeClustering(n_clusters=4)
agg.fit(x_scaled)
```

Out[ ]:
```
▾ AgglomerativeClustering
AgglomerativeClustering(n_clusters=4)
```

In [ ]:
```python
kmc_score = silhouette_score(x_scaled, kmc.labels_, metric = 'euclidean')
agg_score = silhouette_score(x_scaled, agg.labels_, metric = 'euclidean')
```

In [ ]:
```python
from sklearn.decomposition import PCA
```

Reducing features to 25 principal components

In [ ]:
```python
pca = PCA(n_components=25)
x_scaled_pca = pca.fit_transform(x_scaled)
```

In [ ]:
```python
scaled_pca_df = pd.DataFrame(x_scaled_pca)
scaled_pca_df.head(10)
```

Out[ ]:

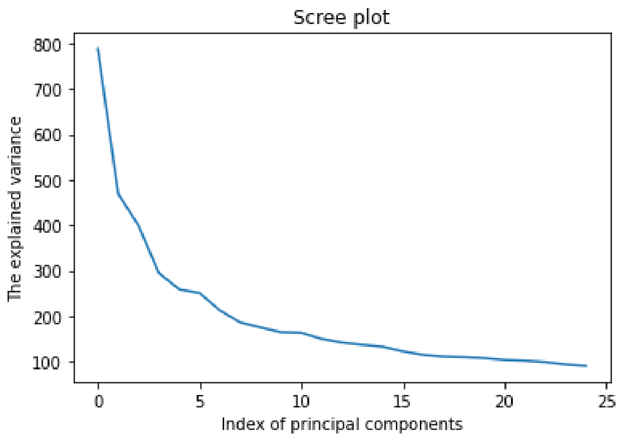
	0	1	2	3	4	5	6	7	8	9	...	
0	-19.837922	-3.556709	-9.813077	-0.829050	12.603045	7.487612	14.184651	-3.187169	22.047830	-20.734430	...	-6.54
1	-23.089265	-6.440586	-13.479945	5.633946	8.039835	3.704302	10.110062	-7.357792	22.412886	-13.837204	...	5.24
2	-27.456089	-2.465385	-3.532241	-1.357396	12.558320	17.350768	10.370434	-2.648433	-0.234753	-6.768361	...	-7.86
3	-42.816808	9.767721	-0.889045	3.442682	42.255995	27.235973	17.557540	-0.530678	14.211893	16.508761	...	-20.69
4	-55.418670	5.200681	-21.096149	15.847014	10.467658	12.970534	12.552774	32.386505	-7.855514	-10.455047	...	-3.84
5	-27.178090	-6.779642	-21.816109	13.844911	-7.991001	0.707358	27.980747	31.244623	-10.915576	1.328298	...	-16.99
6	-31.446156	-3.862969	-30.352621	41.688820	-10.412839	-17.011885	23.724993	-0.915725	14.063038	-8.255355	...	14.30
7	-22.332538	-10.395151	-18.755855	6.974779	5.542044	11.705361	11.761207	22.794015	-3.752483	-5.006886	...	9.96
8	-14.289788	-16.111027	-19.758178	6.576967	3.781135	-8.011857	-13.098951	7.209085	0.912752	-8.181110	...	-6.35
9	-29.748111	-23.993437	-5.884051	-10.014191	-3.450814	11.706664	0.557858	8.059077	-20.051653	-27.663877	...	12.29

10 rows × 25 columns

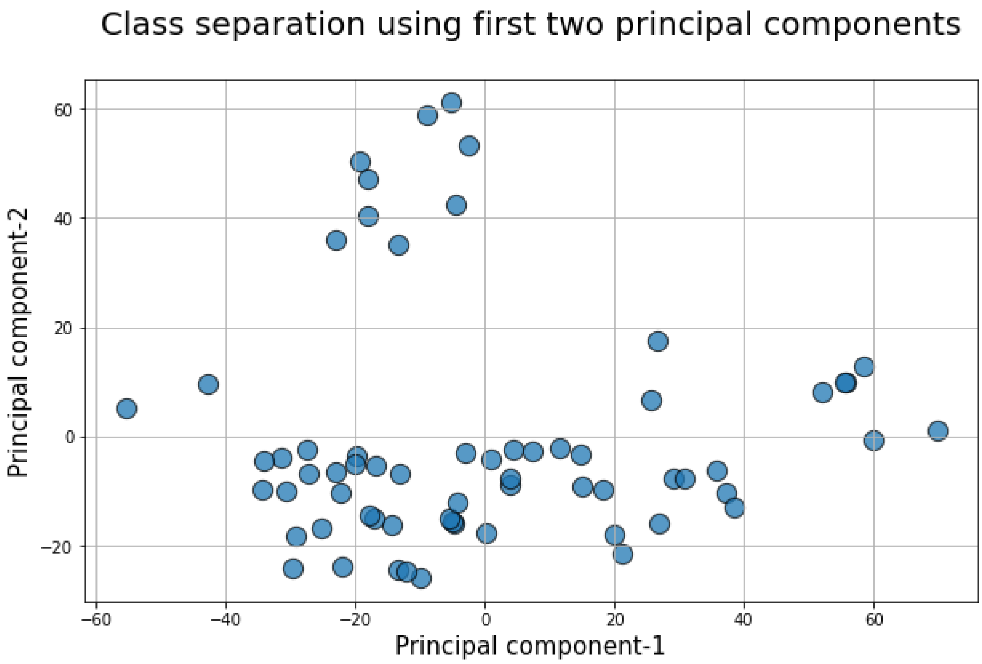
In [ ]:
```python
print(pca.explained_variance_ratio_)
print(pca.explained_variance_ratio_.sum())
```

```
[0.11358942 0.06756202 0.05751842 0.04247547 0.03734964 0.03618621
 0.03066211 0.02685837 0.0252933  0.02375098 0.0235542  0.02163616
 0.02047736 0.01976946 0.01914633 0.01769217 0.01659276 0.01605762
 0.01588125 0.01557134 0.01497665 0.01474998 0.01428692 0.01356469
 0.01309162]
0.7182944736449139
```

In [ ]:
```python
ax = sns.lineplot(pca.explained_variance_)
ax.set_title("Scree plot")
ax.set_xlabel("Index of principal components")
ax.set_ylabel("The explained variance")
plt.show()
```



In [ ]:
```python
plt.figure(figsize=(10,6))
plt.scatter(x_scaled_pca[:,0],x_scaled_pca[:,1],edgecolors='k',alpha=0.75,s=150)
plt.grid(True)
plt.title("Class separation using first two principal components\n",fontsize=20)
plt.xlabel("Principal component-1",fontsize=15)
plt.ylabel("Principal component-2",fontsize=15)
plt.show()
```



creating 4 kmeans clusters on pca reduced dataset

In [ ]:
```
kmc_pca = KMeans(n_clusters=4)
kmc_pca.fit(x_scaled_pca)
```

Out[ ]:
```
▾ KMeans
KMeans(n_clusters=4)
```

creating 4 agglomerative clusters on pca reduced dataset

In [ ]:
```
agg_pca = AgglomerativeClustering(n_clusters=4)
agg_pca.fit(x_scaled_pca)
```

Out[ ]:
```
▾ AgglomerativeClustering
AgglomerativeClustering(n_clusters=4)
```

In [ ]:
```
kmc_score_pca = silhouette_score(x_scaled_pca, kmc_pca.labels_, metric = 'euclidean')
agg_score_pca = silhouette_score(x_scaled_pca, agg_pca.labels_, metric = 'euclidean')
```

Comparing scores

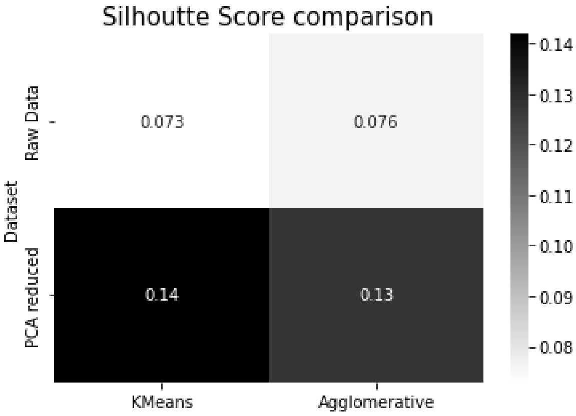
In [ ]:
```
comparison = (('Raw Data', kmc_score, agg_score),
      ('PCA reduced', kmc_score_pca, agg_score_pca))
```

In [ ]:
```
comparison_df = pd.DataFrame(comparison, columns = ["Dataset", "KMeans", "Agglomerative"])
comparison_df.set_index('Dataset', inplace=True)
comparison_df
```

Out[ ]:

	KMeans	Agglomerative
Dataset		
Raw Data	0.072993	0.076382
PCA reduced	0.141810	0.127651

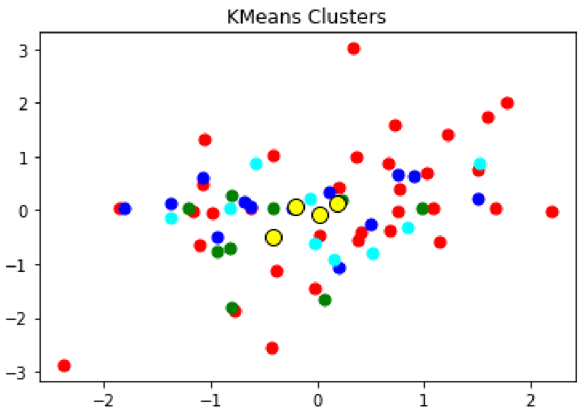
In [ ]:
```
sil_comparison = sns.heatmap(comparison_df,cmap='binary', annot=True)
sil_comparison.set_title("Silhoutte Score comparison",fontsize=15)
plt.show()
```



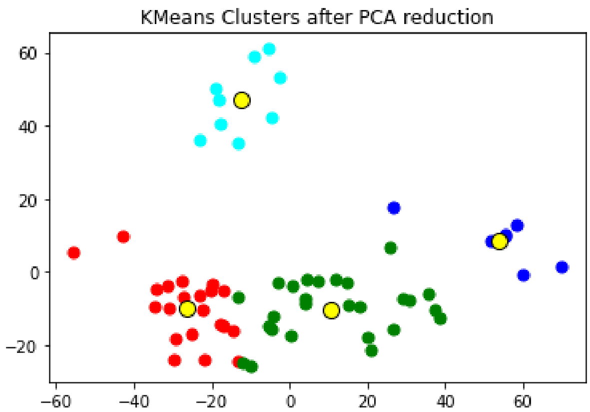
Therefore K means clustering on PCA reduced dataset gives the best score

Plotting Comparison

In [ ]:
```
y_kmc = kmc.fit_predict(x_scaled)
plt.scatter(x_scaled[y_kmc==0, 0], x_scaled[y_kmc==0, 1], s=50, c='red', label ='KMC Cluster 1')
plt.scatter(x_scaled[y_kmc==1, 0], x_scaled[y_kmc==1, 1], s=50, c='blue', label ='KMC Cluster 2')
plt.scatter(x_scaled[y_kmc==2, 0], x_scaled[y_kmc==2, 1], s=50, c='green', label ='KMC Cluster 3')
plt.scatter(x_scaled[y_kmc==3, 0], x_scaled[y_kmc==3, 1], s=50, c='cyan', label ='KMC Cluster 4')
#Plot the centroid. This time we're going to use the cluster centres  #attribute that returns here the coordina
plt.scatter(kmc.cluster_centers_[:, 0], kmc.cluster_centers_[:, 1], s=100, c='yellow',edgecolors='k', label =
plt.title('KMeans Clusters')
plt.show()
```

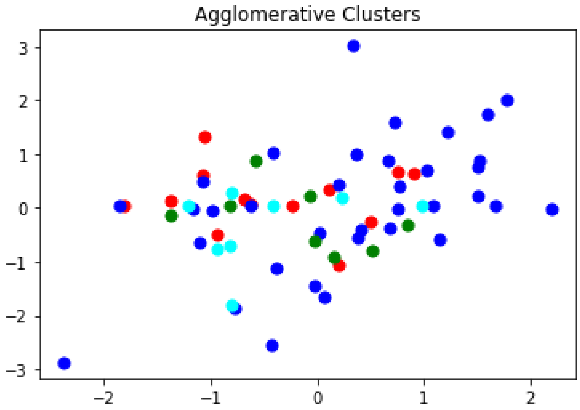


In [ ]:
```
y_kmc_pca = kmc_pca.fit_predict(x_scaled_pca)
plt.scatter(x_scaled_pca[y_kmc_pca==0, 0], x_scaled_pca[y_kmc_pca==0, 1], s=50, c='red', label ='KMC PCA Cluste
plt.scatter(x_scaled_pca[y_kmc_pca==1, 0], x_scaled_pca[y_kmc_pca==1, 1], s=50, c='blue', label ='KMC PCA Clust
plt.scatter(x_scaled_pca[y_kmc_pca==2, 0], x_scaled_pca[y_kmc_pca==2, 1], s=50, c='green', label ='KMC PCA Clus
plt.scatter(x_scaled_pca[y_kmc_pca==3, 0], x_scaled_pca[y_kmc_pca==3, 1], s=50, c='cyan', label ='KMC PCA Clust
#Plot the centroid. This time we're going to use the cluster centres  #attribute that returns here the coordina
plt.scatter(kmc_pca.cluster_centers_[:, 0], kmc_pca.cluster_centers_[:, 1], s=100, c='yellow',edgecolors='k', l
plt.title('KMeans Clusters after PCA reduction')
plt.show()
```



In [ ]:
```
y_agg = agg.fit_predict(x_scaled)
plt.scatter(x_scaled[y_agg==0, 0], x_scaled[y_agg==0, 1], s=50, c='red', label ='Agg Cluster 1')
plt.scatter(x_scaled[y_agg==1, 0], x_scaled[y_agg==1, 1], s=50, c='blue', label ='Agg Cluster 2')
plt.scatter(x_scaled[y_agg==2, 0], x_scaled[y_agg==2, 1], s=50, c='green', label ='Agg Cluster 3')
plt.scatter(x_scaled[y_agg==3, 0], x_scaled[y_agg==3, 1], s=50, c='cyan', label ='Agg Cluster 4')
#Plot the centroid. This time we're going to use the cluster centres  #attribute that returns here the coordina
#plt.scatter(agg.cluster_centers_[:, 0], agg.cluster_centers_[:, 1], s=100, c='yellow',edgecolors='k', label =
```

```
plt.title('Agglomerative Clusters')
plt.show()
```



In [ ]:

```
y_agg_pca = agg_pca.fit_predict(x_scaled_pca)
plt.scatter(x_scaled_pca[y_agg_pca==0, 0], x_scaled_pca[y_agg_pca==0, 1], s=50, c='red', label ='Agg PCA Cluste
plt.scatter(x_scaled_pca[y_agg_pca==1, 0], x_scaled_pca[y_agg_pca==1, 1], s=50, c='blue', label ='Agg PCA Clust
plt.scatter(x_scaled_pca[y_agg_pca==2, 0], x_scaled_pca[y_agg_pca==2, 1], s=50, c='green', label ='Agg PCA Clus
plt.scatter(x_scaled_pca[y_agg_pca==3, 0], x_scaled_pca[y_agg_pca==3, 1], s=50, c='cyan', label ='Agg PCA Clust
#Plot the centroid. This time we're going to use the cluster centres  #attribute that returns here the coordina
#plt.scatter(agg_pca.cluster_centers_[:, 0], agg_pca.cluster_centers_[:, 1], s=100, c='yellow',edgecolors='k',
plt.title('Agglomerative Clusters after PCA reduction')
plt.show()
```